Distribution Category:
Mathematics and
Computer Science (UC-405)

ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439-4801

ANL--91/1

DE91 007145

ANL-91/1

CELEFUNT: A Portable Test Package for Complex Elementary Functions

by

W. J. Cody

Mathematics and Computer Science Division

January 1991

This work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U. S. Department of Energy, under Contract W-31-109-Eng-38.

Contents

CELEFUNT: A Portable Test Package for Complex Elementary Functions

by

W. J. Cody

Abstract

This paper discusses CELEFUNT, a package of Fortran programs for testing complex elementary functions.

1 Introduction

CELEFUNT is a collection of test programs for the complex floating-point elementary functions required by the 1978 ANSI Fortran Standard (CABS, CSQRT, CLOG, CEXP, CSIN/CCOS, and the complex power function) [1]. It is a companion to the ELEFUNT package [3] introduced over ten years ago for testing the real floating-point elementary functions and a forthcoming INTFUNT package for testing intrinsic functions involving integers. These packages, together with the environmental inquiry programs MACHAR [2] and PARANOIA [4], provide the means for a thorough examination of the computational environment available to the Fortran programmer.

The next section provides a broad overview of the techniques used in CELEFUNT to assess accuracy. Section 3 discusses portability issues, while subsequent sections discuss the individual test programs in more detail. Each discussion includes an error analysis, a tabulation of test results on three different systems (SUN Fortran 1.2 under SUNOS 4.1 running on a Sun 3/60, VAX Fortran under VMS 5.3 running on a VAX 8700, and Lahey F77L 4.0 running under DOS 3.30 on an IBM/XT with a coprocessor), and an interpretation of those results to aid others in interpreting their results.

We standardize our notation in the following discussions so that mathematical quantities are expressed in lower case (x, $sin(x)$, etc.) and machine quantities are expressed in upper case (X, $SIN(X)$, etc.). Where the distinction is not important, we will use the mathematical notation.

2 Overview of Testing

There are three widely accepted approaches to testing the accuracy of function routines. In order of decreasing resolution these are table-driven techniques [5, 6], comparison with higher-precision computations, and evaluation of carefully selected identities. The first of these is capable of determining the error in a function evaluation to within a small fraction (typically 0.001 or less) of an ulp (unit in the last place) and is by far the preferred method. Programs implementing this method are portable but require great care in preparation. They also usually require lengthy machine runs for each test; however, because these tests need be run infrequently, this latter characteristic is not a serious drawback. The second method is capable of determining errors to within 0.5 to 1 ulp, depending on the arithmetic characteristics of the host machine. The major disadvantage of this approach is that testing of functions written for the highest precision arithmetic on the machine requires an inefficient extended-precision arithmetic package. Programs for testing-lower precision functions either use the same arithmetic package (increasing portability at the expense of efficiency) or use the native higher-precision arithmetic (increasing efficiency at the expense of portability). The third approach is generally capable of determining errors to within less than 2 ulps, and often to within 1 ulp, depending upon the circumstances of the test (the identity used, domain tested, etc.). Test programs implementing this approach are again highly portable.

We regard the first and third approaches as being the most practical. The use of identities is usually adequate for distinguishing between acceptable and unacceptable function programs, but table-driven methods are essential to identify truly exceptional function programs.

CELEFUNT relies on identities to determine the accuracy of seven different complex elementary functions required by the 1978 Fortran standard. The general approach is to select an identity that involves only one or two evaluations of the function under test and that is numerically stable (i.e., evaluation of the identity does not introduce serious contamination of the error being measured) over some region of interest. Because we are testing complex-valued functions, we are interested in three different error measurements: the error as a complex result, and the error in each of the components of the result. Identities numerically stable for one of these measurements may not be stable for all three; hence, some care is needed in the choice of the identities and of the test regions. Details of these choices are discussed for each of the test programs a little later.

Given an identity and a test domain, the procedure is to select a reasonable number M (we use 2,000) of test arguments from a uniform random distribution over the region. We divide the domain of the real component into M subdomains, selecting test arguments by taking a real component from each of the subdomains in turn, and pairing it with a random imaginary component drawn from the undivided domain of imaginary components. Where the test domain is not rectangular, we use a rejection procedure on the imaginary component. Clearly this approach does not provide a truly uniform distribution in nonrectangular domains, but we do believe that the deviation from uniformity in these cases is not detrimental to our tests.

Once the test argument has been selected, it is *purified* if necessary to guarantee that it and related derived arguments are all exact machine numbers. For example, the test of CABS uses the arguments $3 \times Z$, $4 \times Z$, and $5 \times Z$, where Z is random. To guarantee that rounding error in the evaluation of these auxiliary arguments does not contaminate the tests, we perturb the last few bits of each component of Z to form a nearby argument $\hat{Z}$ such that the products $3 \times \hat{Z}$, $4 \times \hat{Z}$, and $5 \times \hat{Z}$ are all exact machine-representable numbers. The details of this purification process necessarily varies from one test to another, but it is a simple process in every case. We then evaluate the expression

$$E = \left| \frac{F(\hat{Z}) - I(\hat{Z})}{F(\hat{Z})} \right|,$$

where $F(\hat{Z})$ is the function evaluation and $I(\hat{Z})$ is the evaluation of the identity. In this case, E is an estimate of the complex relative error in $F(\hat{Z})$. Analogous expressions are used to evaluate the component errors.

Finally, we gather error statistics for the test. The statistics reported for each region includes N, the number of times $E = 0$; MRE, the maximum relative error as measured by E; RMS, the root mean square error; and the $\hat{Z}$ and $F(\hat{Z})$ corresponding to MRE. These latter quantities are helpful for analysis of the computation leading to the MRE.

Both MRE and RMS are reported as an estimated loss of base-β significant digits, where β is the floating-point radix. This measurement is independent of the machine wordlength and hence is useful for comparing results on different machines with the same β. The equations used are

$$\mathrm{MRE} = \max[0.0, p + \ln(\max|E|)/\ln(\beta)],$$

and

$$\mathrm{RMS} = \max[0.0, p + \ln(\textstyle\sum E^2/M)/(2\ln(\beta))],$$

where p is the number of significant base-β digits in the significand. Note that the computation of MRE and RMS has been adjusted so they never report a negative loss of significant digits. With these definitions the complex relative error E may not be zero even though the MRE is. This situation can occur when the erroneous component of $F(\hat{Z})$ is much smaller in magnitude than the correct component. Also note that for $M = 2000$ and $\beta = 2$, $RMS \geq MRE - 5.48$.

Accuracy tests based on identities are augmented with additional small tests looking at the preservation of mathematical properties, such as $\exp(x) \times \exp(-x) = 1$, and the behavior with extreme arguments. Auxiliary tests of this type are grouped at the end of the program in increasing order of probability of an exception that might terminate program execution.

When the test program is finished, we attempt to determine experimentally how accurate it is by *calibrating* it. Calibration consists of running the test program in single-precision arithmetic with a function program that accepts single-precision arguments, does all computations in double precision, and then returns single-precision results. Such a function program returns values correct to within the rounding error on the machine. Thus, the errors reported by the test program represent the "noise" in the testing process; they measure the ability of the test program to detect "perfection."

3 Portability Issues

The programs in this package have been written with portability in mind. To this end, we impose strong typing; that is, every variable is declared in a Fortran TYPE statement. Although the Fortran standard does not include a specification for double-precision complex arithmetic, most popular compilers today provide that capability. A brief survey showed that more compilers recognize the data type COMPLEX*16 than the data type DOUBLE COMPLEX, with some recognizing both. We have therefore used the former syntax in our programs. This must be altered for those few compilers that require a different declaration.

All floating-point or complex constants are initialized in DATA statements to localize the changes that must be made in moving from single- to double-precision versions of the programs. Conversion from one data type to another is explicit for the same reason, and generic names are used for the intrinsic and elementary functions.

Where statements must be modified (DATA and TYPE statements, for example) to change working precision, alternative statements containing the modifications are provided. Those statements needed for a single-precision version of the program are identified with the characters "CS" in the first two columns, those needed for a double-precision version are identified with the characters "CD," and those needed for a calibration run are identified with the characters "CC." Thus, a global replacement of "CS" in columns 1 and 2 with blanks prepares a single-precision version of the test program, while a global replacement of "CD" with blanks prepares a double-precision version. Replacing both "CS" and "CC" with blanks prepares a calibration version complete with the necessary auxiliary function that does internal computations in double precision. Note that the programs will not compile correctly without one of these global changes.

Finally, the package contains both a random number generator and the environmental inquiry program MACHAR to determine the necessary machine-dependent parameters. MACHAR is known to malfunction on a few machines (see [2]). On those machines, the test programs must be modified to delete the call to MACHAR and to provide the necessary machine parameter values in some other way, perhaps in DATA statements. The comments at the top of each test program define the needed parameters as an aid in making such changes.

Many of the test programs and MACHAR contain computations that are sensitive to the order of evaluation and, more important, to the precision of intermediate results. These programs have been carefully written to avoid trouble, but they may malfunction if compiler optimization alters the order of evaluation or if it retains and uses results in higher-than-working precision. It is imperative that compiler optimization be turned off.

Be warned that the random number program, REN, has not been tested for general-purpose generation of random numbers. It is adequate for our use, but we cannot attest to its suitability for any other purpose. As with the test programs, appropriate versions of both REN and MACHAR must be prepared with global editing.

We believe these programs to be completely portable between machines and precisions subject to the above comments.

4 Test of CABS

Let $z = x + iy$. Then $|z| = \sqrt{x^2 + y^2}$ maps the complex plane onto the positive real axis. Our accuracy tests for CABS, which exploit this definition, are unusual in that they are completely free of any extraneous error. The first identity used is based on the Pythagorean number set (3,4,5). Given a random machine number X drawn from the interval (1,20), our program perturbs X in the low-order bits to obtain V, say, so that $3V$, $4V$, and $5V$ are all exactly representable machine numbers. This is accomplished by the Fortran statements

```
Y = X * EIGHT
Z = X + Y
V = Z - Y
```

which zeroes out the last four bits of X, provided all assignments to the left-hand variables are at working precision. Because the significand of 5 is exactly representable in three bits, and the number of significant bits of a product is at most the sum of the significant bits in the factors, all of the desired products with V are exact machine numbers (the extra trailing zero bit in V protects against the lack of a guard bit, improper rounding, etc.). Then

Table 1: **Test Results for CABS**

Test/Machine	N	MRE	RMS
$\lvert(3x,4x)\rvert$ **vs** $5x$			
$x \in (1,20)$			
Sun 3/60 Cal./SP/DP	2000	0.00	0.00
IBM XT, Lahey Cal./SP/DP	2000	0.00	0.00
VAX, Cal./SP/G-DP	2000	0.00	0.00
$\lvert(5x,12x)\rvert$ **vs** $13x$			
$x \in (1,20)$			
Sun 3/60 Cal./SP/DP	2000	0.00	0.00
IBM XT, Lahey Cal./SP/DP	2000	0.00	0.00
VAX, Cal.	2000	0.00	0.00
VAX, SP	1246	0.30	0.00
VAX, G-DP	0	1.00	0.51

$$E = \frac{CABS(W) - 5V}{CABS(W)},$$

where $W = (3+4i)V$, estimates the error in $CABS(W)$. That is, if we let $CABS(W) = |w|(1+\delta)$,

$$E = \frac{|w|(1+\delta) - 5v}{|w|(1+\delta)}.$$

Multiplying numerator and denominator by $(1-\delta)$ and retaining only first-order error terms, we obtain

$$E = \delta.$$

The second test is identical to the first, except that it uses the Pythagorean number set (5,12,13) and the last five bits of X are zeroed out. While these tests are limited to two different rays in the complex plane, we believe that the test results are representative of the function behavior everywhere. If doubt exists, it is easy to modify or augment the tests to use other Pythagorean number sets or to draw X from other intervals.

The auxiliary tests include invocation of CABS for the extreme arguments $Z = (3+4i)XMIN$, $W = (5/16 + 12/16i)XMAX$, and $Z = 4W$, where $XMIN$ is the smallest positive normalized floating-point number and $XMAX$ is the largest finite floating-point number. The first two cases should not cause trouble, while the final computation should provoke an error return of some sort.

Table 1 presents the results of running our test program on three representative systems. The calibration runs found no errors at all, corroborating our assertion that the test procedure was itself error-free. Indeed, the first test never detected an error on any of the machine/precision combinations tested. Although the VAX routines were not perfect in the second test, the maximum relative error detected was only 1 bit, indicating that the programs are accurate to within rounding error. The routines all passed the auxiliary tests, with the SUN programs returning a value of ∞ in the last test, and the other two systems aborting execution with error messages that did not identify CABS as the culprit. Overall, these CABS programs appear to be superb.

5 Test of CSQRT

Represent z in polar coordinates, $z = \rho e^{i\theta}$. Then $\sqrt{z} = \sqrt{\rho} + \theta/2$ maps the complex plane slit along the negative real axis onto the right half of the complex plane.

Accuracy tests of CSQRT are based on the identity $\sqrt{z^2} = z$ applied over the square region with vertices at (0,0), and (10,10), and again over the square region with vertices at (0,0) and $(-100, 100)$. The algebraic sign of the result must be changed in the second case.

Roundoff error in the computation of $Z * Z$ is minimized by purifying both X and Y, the real and imaginary components of Z, so that slightly more than half of the trailing bits are zero in each case and the products $X * X$, $Y * Y$, and $X * Y$ are all exact machine numbers. Then $Z * Z$ is explicitly constructed with real component $X * X - Y * Y$ and imaginary component $2 * X * Y$. Roundoff error is limited to at most one rounding error in the real component, and the imaginary component is completely free of error.

The magnitude of the complex error in CSQRT is estimated as the absolute value of

$$E = \frac{\mathrm{CSQRT}(Z * Z) - \mathrm{Z}}{\mathrm{CSQRT}(Z * Z)}.$$

Let $\mathrm{CSQRT}(Z * Z) = \sqrt{(z^2)(1+\epsilon)}(1+\delta)$, where δ is the relative error in the CSQRT function and ϵ is the relative error in evaluating $Z * Z$. Because the arguments have been purified, ϵ is a real quantity, the error in the real component of $Z * Z$. Then

$$E = \frac{\sqrt{z^2(1+\epsilon)}(1+\delta) - z}{\sqrt{z^2(1+\epsilon)}(1+\delta)}.$$

If one ignores higher-order terms in the errors, this simplifies to

$$E = \delta + \epsilon/2.$$

Thus E contaminates δ, the error we wish to measure, with half the rounding error in the evaluation of the real component of $Z * Z$. We expect this contamination to be small and the statistics based on E to be a reliable indicator of the accuracy of CSQRT.

To measure the error in the real component of $\mathrm{CSQRT}(Z * Z)$, set

$$\hat{E} = \frac{\mathrm{REAL}\{\mathrm{CSQRT}(Z * Z)\} - X}{\mathrm{REAL}\{\mathrm{CSQRT}(Z * Z)\}}.$$

Letting $\hat{\delta}$ be the relative error in the real component of $\mathrm{SQRT}(Z * Z)$, we have

$$\hat{E} = \frac{\Re\{\sqrt{z^2(1+\epsilon)}\}(1+\hat{\delta}) - x}{\Re\{\sqrt{z^2(1+\epsilon)}\}(1+\hat{\delta})}.$$

Table 2: **Test Results for CSQRT**

					Error in			Error in	
Test/Machine	Vector Error			Real Component			Imag. Component		
	N	MRE	RMS	N	MRE	RMS	N	MRE	RMS
$\sqrt{(z \times z)}$ **vs** z $z \in (0,10) \times (0,10)$									
Sun 3/60 Cal./SP/DP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
Sun 3/60 DP	1999	0.00	0.00	1999	0.00	0.00	1999	0.14	0.00
IBM XT, Lahey Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
IBM XT, Lahey DP	1999	0.00	0.00	1999	9.46	3.98	2000	0.00	0.00
VAX, Cal.	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, SP	1826	1.17	0.00	1826	1.39	0.00	1826	1.00	0.00
VAX, G-DP	1172	1.16	0.00	1172	1.37	0.01	1172	1.38	0.02
$-\sqrt{(z \times z)}$ **vs** z $z \in (0,-100) \times (0,100)$									
Sun 3/60 Cal./SP/DP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
IBM XT, Lahey Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
IBM XT, Lahey DP	1998	0.00	0.00	1999	0.27	0.00	1999	0.01	0.00
VAX, Cal.	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, SP	1836	1.00	0.00	1836	1.00	0.00	1836	1.01	0.00
VAX, G-DP	1201	1.14	0.00	1201	1.34	0.00	1201	1.36	0.00

Again ignoring higher-order terms in the errors, we can simplify this to

$$\hat{E} = \hat{\delta} + \epsilon/2.$$

Similarly,

$$\tilde{E} = \tilde{\delta} + \epsilon/2$$

measures the error in the imaginary component of CSQRT($Z * Z$). These expressions show that $\hat{E}$ and $\tilde{E}$ are reasonable estimates of the component errors.

The calibration results presented in Table 2 verify the quality of our test program, and the other results indicate the overall high quality of the programs tested. The large error detected in the real component in the first double-precision test under the Lahey compiler on the XT occurred for a Y over 3,000 times larger than X, hence for $Z * Z$ large and close to the negative real axis. Because all other sources of error have been eliminated, and all of the other programs tested handled this same case with little or no error, we suspect that Lahey's CSQRT has a problem computing results very close to the imaginary axis.

The special argument tests uncovered important problems with the double-precision program on the Sun and both programs on the VAX. The Sun program returned a real component of ∞

and all of the VAX tests failed with an overflow error message for the argument $(1+i)XMAX$, where $XMAX$ is the largest floating-point number. Curiously, the programs functioned correctly for the argument $(1+i)XMIN$, where $XMIN$ is the smallest positive normalized floating-point number. The obvious programming mistake should have given an underflow message in the latter case, so the mistake is more subtle.

6 Test of CLOG

Again represent z in polar coordinates, $z = \rho e^{i\theta}$. Then $\ln z = \ln \rho + i\theta$ maps the complex plane slit along the negative real axis onto the infinite strip $|y| \leq \pi$, with the unit circle mapping onto the imaginary axis in that strip.

The accuracy tests of CLOG are based on the identity $\ln z = \ln(z^2)/2$ applied over the rectangular regions with vertices at (2,0) and (10,10), (1C00,–1000) and (2000,–4000), and $(\varepsilon,-\varepsilon)$ and (0.25,–0.25), where ε is of the order of roundoff error on the machine. For these tests, arguments are purified in the same way as for the tests of CSQRT, so the imaginary component of $Z*Z$ is exact. Because none of our test regions include the unit circle, the real part of $\ln(z)$ never vanishes.

The magnitude of the complex error in CLOG is estimated as the absolute value of

$$E = \frac{\mathrm{CLOG}(Z) - \mathrm{CLOG}(Z*Z)/2}{\mathrm{CLOG}(Z)}.$$

Let δ be the complex relative error in CLOG(Z), ϵ be the relative error in evaluating $Z*Z$, and σ be the relative error in evaluating CLOG($Z*Z$). Then

$$E = \frac{\ln(z)(1+\delta) - \ln(z^2[1+\epsilon])(1+\sigma)/2}{\ln(z)(1+\delta)},$$

which reduces to

$$E = \delta - \sigma - \frac{\epsilon}{2\ln(z)}$$

when higher-order terms in the error are neglected. The corresponding estimate for error in the real component is

$$\hat{E} = \frac{\Re[\ln(z)](1+\hat{\delta}) - \Re\{\ln(z^2[1+\epsilon])\}(1+\hat{\sigma})/2}{\Re\{\ln(z)(1+\hat{\delta})\}},$$

which reduces to

$$\hat{E} = \hat{\delta} - \hat{\sigma} - \frac{\epsilon}{2 \times \Re\{\ln(z)\}}.$$

The reduced expression for the error in the imaginary component,

$$\tilde{E} = \tilde{\delta} - \tilde{\sigma},$$

is even simpler, because ϵ is real.

Table 3: **Test Results for CLOG**

Test/Machine	Vector Error			Error in Real Component			Error in Imag. Component		
	N	MRE	RMS	N	MRE	RMS	N	MRE	RMS
$\ln(z)$ vs $\ln(z^2)/2$									
$z \in (2,10) \times (0,10)$									
Sun 3/60 Cal./SP	1988	0.00	0.00	2000	0.00	0.00	1988	0.95	0.00
Sun 3/60 DP	1696	0.98	0.00	1724	1.00	0.00	1963	0.97	0.00
IBM XT, Lahey Cal./SP	1988	0.00	0.00	2000	0.00	0.00	1988	0.95	0.00
IBM XT, Lahey DP	1978	0.86	0.00	1994	0.86	0.00	1980	0.95	0.00
VAX, Cal.	1988	0.00	0.00	2000	0.00	0.00	1988	0.95	0.00
VAX, SP	983	1.46	0.00	1573	1.47	0.00	1248	1.30	0.00
VAX, G-DP	1085	0.99	0.00	1524	1.00	0.00	1406	1.74	0.00
$z \in (1000, 2000) \times$ $(-1000, -4000)$									
Sun 3/60 Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
Sun 3/60 DP	1908	0.99	0.00	1931	1.00	0.00	1975	0.96	0.00
IBM XT, Lahey Cal./SP/DP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, Cal.	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, SP	981	1.00	0.00	1883	1.00	0.00	1044	1.28	0.12
VAX, G-DP	1395	0.99	0.00	1886	1.00	0.00	1475	0.99	0.00
$z \in (\varepsilon, 0.25) \times (-\varepsilon, -0.25)$									
Sun 3/60 Cal./SP	1988	0.41	0.00	1996	0.41	0.00	1992	0.94	0.00
Sun 3/60 DP	1648	0.97	0.00	1673	1.00	0.00	1969	0.97	0.00
IBM XT, Lahey Cal./SP	1988	0.41	0.00	1996	0.41	0.00	1992	0.94	0.00
IBM XT, Lahey DP	1973	0.92	0.00	1987	0.93	0.00	1982	0.96	0.00
VAX, Cal.	1988	0.41	0.00	1996	0.41	0.00	1992	0.94	0.00
VAX, SP	894	1.46	0.00	1455	1.69	0.00	1209	1.30	0.00
VAX, G-DP	972	1.38	0.00	1390	1.56	0.00	1392	1.35	0.00

In all of these expressions, the first two error terms are inherent to E, which involves two evaluations of CLOG. Coefficients for the terms involving ϵ involve division by twice some component of $\ln(z)$. The selection of test regions guarantees that these coefficients are less than 1 in magnitude. For example, in the first test $0.693 < \ln 2 \leq \Re \ln z \leq |\ln z|$, so the coefficients of ϵ in E and $\hat{E}$ are bounded above by 0.722.

The calibration results for these tests presented in Table 3 show that the testing error is bounded by one bit. Note in particular that the calibration results are identical for all three systems tested, attesting to the validity of these results across systems. We see nothing in the tabulated results for the system-supplied routines to indicate problems.

As with the CSQRT tests, the double-precision program on the Sun and both programs on the VAX malfunctioned for the argument (XMAX,XMAX). The Sun program returned a real component of ∞, and all of the VAX tests failed with an overflow error message.

7 Test of CEXP

Let $z = x + iy$. Then $\exp z = \exp(x)(\cos y + i \sin y)$ is a periodic function of period $2\pi i$. It maps the infinite horizontal strip $|y| \le \pi$ into the complex plane slit along the negative real axis. The real axis is mapped onto the positive real axis, the imaginary axis within the strip is mapped onto the unit circle, and the lines $|y| = \pi$ are mapped onto the slit.

Our tests of CEXP exploit the identity $\exp(z) = \exp(z - \Delta)\exp(\Delta)$, $\Delta = (1 + i)/16$, over appropriate regions. Should the real or imaginary components of $\exp(z)$ be significantly less in magnitude than the corresponding components of $\exp(z - \Delta)$, the product $\exp(z - \Delta)\exp(\Delta)$ must involve the subtraction of nearly equal quantities and hence a large cancellation error. To minimize this problem, we choose regions that ιest the exponential dependence on x while staying away from regions in which $\sin(y)$ or $\cos(y)$ vanishes or even approaches a zero with increasing y. The rectangles with vertices at (0,0.0625) and (1,1), (1.625,1.625) and (3,3), and (16,16) and (17,17) meet these requirements.

The magnitude of the complex error in CEXP is estimated as the absolute value of

$$E = \frac{\text{CEXP}(Z) - \text{CEXP}(Z - DEL)(1 + CDEL)}{\text{CEXP}(Z)},$$

where the term $1 + CDEL$ represents $\exp(\Delta)$ to several decimal places beyond working precision. Errors in this expression are controlled in two ways. First, the arguments are purified so both Z and $Z - DEL$ are exact machine numbers, and second, the product is computed as

$$\text{CEXP}(Z - DEL) + \text{CEXP}(Z - DEL)CDEL,$$

reducing the rounding error to a level that can be ignored. Remaining errors in E are δ, the complex relative error in CEXP(Z), and σ, the relative error in evaluating CEXP($Z - DEL$). Thus

$$E = \frac{\exp(z)(1 + \delta) - \exp(z - \Delta)\exp(\Delta)(1 + \sigma)}{\exp(z)(1 + \delta)}.$$

If only first-order error terms are retained, this becomes

$$E = \delta - \sigma.$$

The expressions for the errors in the real and imaginary components are analogous.

Table 4 summarizes the test results. Note that the calibration runs on the three systems are in general agreement, and that the MRE measured in the calibration runs is consistent with our analysis. Based on the tabulated results, the CEXP routines tested look good.

Table 4: **Test Results for CEXP**

Test/Machine	Vector Error			Error in Real Component			Error in Imag. Component		
	N	MRE	RMS	N	MRE	RMS	N	MRE	RMS
e^z **vs** $e^{z-\delta}e^\delta$ $\delta = (1+i)/16$									
$z \in (0,1) \times (1/16,1)$									
Sun 3/60 Cal./SP	1121	0.98	0.00	1512	1.00	0.00	1479	1.02	0.00
Sun 3/60 DP	539	1.96	0.15	1010	2.00	0.17	995	1.93	0.18
IBM XT, Lahey Cal./SP	1129	0.97	0.00	1489	0.99	0.00	1513	1.00	0.00
IBM XT, Lahey DP	1127	0.99	0.00	1495	1.00	0.00	1505	1.00	0.00
VAX, Cal.	1150	0.98	0.00	1502	1.00	0.00	1528	1.00	0.00
VAX, SP	574	1.85	0.11	1077	1.90	0.12	985	1.98	0.17
VAX, G-DP	530	1.74	0.14	1023	1.99	0.15	982	1.97	0.21
$z \in (1.625,3) \times (1.625,3)$									
Sun 3/60 Cal./SP	1069	0.97	0.00	1483	1.06	0.00	1449	1.22	0.00
Sun 3/60 DP	558	1.73	0.11	1061	1.95	0.10	1000	2.03	0.19
IBM XT, Lahey Cal./SP	1116	0.98	0.00	1528	0.99	0.00	1453	1.22	0.00
IBM XT, Lahey DP	1155	0.97	0.00	1546	1.00	0.00	1477	1.18	0.00
VAX, Cal.	1105	0.98	0.00	1529	1.10	0.00	1457	1.10	0.00
VAX, SP	541	1.75	0.12	1012	1.93	0.14	1036	2.08	0.17
VAX, G-DP	515	1.91	0.14	998	1.95	0.19	981	1.96	0.18
$z \in (16,17) \times (16,17)$									
Sun 3/60 Cal./SP	1093	0.92	0.00	1469	1.00	0.00	1490	1.00	0.00
Sun 3/60 DP	518	1.86	0.16	985	1.90	0.21	1016	1.98	0.15
IBM XT, Lahey Cal./SP	1117	0.92	0.00	1471	1.00	0.00	1523	1.00	0.00
IBM XT, Lahey DP	1103	0.97	0.00	1476	1.02	0.00	1495	1.00	0.00
VAX, Cal.	1115	0.92	0.00	1467	1.05	0.00	1520	1.00	0.00
VAX, SP	526	1.83	0.18	974	2.17	0.23	1019	1.98	0.16
VAX, G-DP	485	1.85	0.21	942	1.98	0.25	957	1.92	0.21

Special tests include a brief check that $\exp(z)\exp(-z) = 1$, tests with special arguments, and tests with arguments with components so extreme that an error return of some sort is warranted. The Sun routines always returned a result. Computations with arguments with large negative real components returned zero results, those with large positive imaginary components (so the computation of the requisite $\sin(y)$ and $\cos(y)$ values makes little sense) proceeded without complaint, and those with large positive real components returned (∞, ∞). The Lahey routines aborted the first case with an overflow(!) message, aborted the second case with a correct error message, and incorrectly returned $(0,0)$ in the last case with no indication of an error. The first two cases were processed without complaint on the VAX, but the third correctly aborted with an overflow message.

8 Test of Complex Power Function

The complex power function is the exponentiation function, $z^w = \exp(w \ln z)$. The obvious algorithm is not the most accurate way to compute this function in the real case [3]. Instead, a self-contained computation that evaluates and uses $w \ln z$ to beyond working precision is best. The corresponding algorithm for the complex case has not been published and is probably not known yet. To achieve the most accuracy, single-precision routines might do all internal computations in double precision, and double-precision routines on machines with IEEE floating-point arithmetic might do all internal computations in extended precision. Routines implementing the mathematical definition in working precision will struggle for accuracy.

Our tests are based on three different identities. The first test compares z^w, $w = 1 + 0i$, against z in the rectangular region with vertices at (1,0) and (10,10). This is the purest test possible, measuring the accuracy of the direct exp(ln) cycle; it should easily distinguish between routines that use extra precision internally and those that do not. The magnitude of the complex error in this test is estimated as the absolute value of

$$E = \frac{Z**W - Z}{Z**W},$$

where $W = (1,0)$. Because all arguments are exact, there is no need for argument purification. The usual error analysis reduces this to

$$E = \delta,$$

where δ is the complex relative error in exponentiation. Error estimates for the real and imaginary components are analogous.

For a binary machine, we can estimate the reported value of the vector MRE as follows. The major contribution to the relative vector error is in the computation of $\exp(\ln \rho)$, where ρ is the modulus of z. Further, the relative error in the real exponential function is roughly the absolute error in its argument. In exponentiation routines that use higher-precision arithmetic internally (such as the routine in our calibration runs), the absolute error in $\ln \rho$ will be negligible. In other routines, the detected MRE in our tests should roughly equal the integer part of $\ln(\rho)$ for ρ the maximum modulus in the test region. This integer is 2, which is representable in 2 bits. We therefore expect that calibration runs and tests of routines exploiting higher-precision arithmetic will find no error in this case and that tests of routines exploiting the definition of exponentiation in working-precision arithmetic will report losses slightly greater than 2 bits.

Results reported in Table 5 completely support this analysis. They suggest that the Lahey routines and the single-precision routine on the Sun use higher-precision arithmetic internally, and that all other routines tested use working-precision arithmetic.

Table 5: **Test Results for Complex Power**

					Error in			Error in	
Test/Machine		Vector Error			Real Component			Imag. Component	
	N	MRE	RMS	N	MRE	RMS	N	MRE	RMS
$z^{(1,0)}$ **vs** z									
$z \in (1,10) \times (0,10)$									
Sun 3/60 Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
Sun 3/60 DP	471	2.02	0.42	818	3.16	0.69	839	2.09	0.43
IBM XT, Lahey Cal./SP/DP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, Cal.	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, SP	378	2.06	0.55	718	3.11	0.82	793	2.33	0.56
VAX, G-DP	365	2.14	0.66	715	3.28	0.90	733	2.42	0.70
$z^{(2,0)}$ **vs** $z * z$									
$z \in (1,10) \times (0,10)$									
Sun 3/60 Cal.	1995	0.95	0.00	1995	0.96	0.00	2000	0.00	0.00
Sun 3/60 SP	1993	0.95	0.00	1993	0.96	0.00	2000	0.00	0.00
Sun 3/60 DP	186	2.65	1.22	403	4.41	1.78	486	3.39	1.27
IBM XT, Lahey Cal./SP	1998	0.82	0.00	1998	0.83	0.00	2000	0.00	0.00
IBM XT, Lahey DP	1997	0.42	0.00	1997	0.43	0.00	2000	0.00	0.00
VAX, Cal.	1997	0.84	0.00	1997	0.85	0.00	2000	0.00	0.00
VAX, SP	149	2.97	1.40	375	5.24	2.06	421	3.65	1.45
VAX, G-DP	135	2.98	1.51	353	4.67	2.03	397	3.36	1.54
z^w **vs** $(z * z)^{w/2}$									
$z, w \in (4,10) \times (4,10)$									
Sun 3/60 Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
Sun 3/60 DP	1792	6.50	3.80	1793	13.03	7.83	1792	16.42	10.94
IBM XT, Lahey Cal./SP	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
IBM XT, Lahey DP	1981	0.66	0.00	1988	4.26	0.00	1993	0.94	0.00
VAX, Cal.	2000	0.00	0.00	2000	0.00	0.00	2000	0.00	0.00
VAX, SP	1222	6.44	4.38	1225	12.74	7.80	1224	12.90	7.86
VAX. G-DP	1294	6.67	4.35	1302	17.80	12.32	1303	16.07	10.60

The second test compares z^w, $w = 2 + 0i$, against z^2 in the same rectangular region less a wedge about the line $x = y$. This is a more difficult test of exponentiation than the first test, but is still simple enough to permit reasonably detailed analysis. We purify Z as in the other tests involving $Z * Z$ so there is no rounding error in forming the imaginary component and minimal error in the real component. The magnitude of the complex error in this test is estimated as the absolute value of

$$E = \frac{Z**W - Z*Z}{Z**W},$$

where $W = (2,0)$. Let δ be the complex relative error in exponentiation, and ϵ the relative rounding error in $Z * Z$. Then

$$E = \frac{(z^w)(1+\delta) - z^2(1+\epsilon)}{(z^w)(1+\delta)}.$$

This simplifies to

$$E = \delta - \epsilon,$$

where again only first-order error terms are retained. The error estimates for the real component is analogous, but the ϵ term disappears in the error estimate for the imaginary component (because of argument purification).

The line $x = y$ maps into the positive imaginary axis in this test. Arguments close to this line can contribute massive unavoidable error to the computation. For such arguments, the ln part of the exponentiation computation results in an imaginary component close to $\pi/4$. Regardless of how accurate this computation is, there is some rounding error in the result (albeit much smaller when done in higher-precision arithmetic than when done in working precision). This component is doubled to a value very close to $\pi/2$ and then used as an argument to the sin function in computing the real component of the complex exp. The relative error in the result is roughly equal to the number of significant figures of agreement between the argument and $\pi/2$. Potentially, the real component could lose all significance in this test (losses of 12 or more bits were encountered during refinement of our test procedures). Because the real component is small in magnitude in comparison to the imaginary component, this large component error does not greatly affect the measured vector error. Nevertheless, we have rejected all test arguments with y within 5% of x, thus eliminating a wedge-shaped region from our test domain. With this restriction, it is not difficult to show that the maximum agreement between the argument to the sin and $\pi/2$ is 5 bits.

We expect the MRE on the real component to report a loss of about 5 bits on binary machines, with the vector MRE significantly better than that. The results in Table 5 again corroborate our analysis, and incidentally strengthen our suspicions about which routines use only working-precision arithmetic.

Our final test corresponds more closely to what might be encountered in practice. It compares z^w against $(z^2)^{w/2}$ for pairs of arguments drawn from the square region with vertices at (4,4) and (10,10). Z is purified as before, and W is purified to guarantee that $W/2$ is also an exact machine number. Then the vector error is estimated as the magnitude of

$$E = \frac{Z**W - (Z*Z)**(W/2)}{Z**W}.$$

Let δ be the complex relative error in $Z**W$, σ the relative error in $(Z*Z)**(W/2)$, and ϵ the relative rounding error in $Z*Z$. Then

$$E = \frac{(z^w)(1+\delta) - \{[z^2(1+\epsilon)]^{w/2}\}(1+\sigma)}{(z^w)(1+\delta)}.$$

Retaining only first-order terms, we can simplify this to

$$E = \delta - \sigma - (w/2)\epsilon.$$

Clearly the $w/2$ factor on ϵ is significant and can dominate all other errors in this expression. Unfortunately, such behavior limits this test to a demonstration of the inherent uncertainty of complex exponentiation; it is useless as a true measure of the quality of exponentiation.

The corresponding estimate for the error in the real component is

$$\hat{E} = \frac{\Re(z^w)(1+\hat{\delta}) - \Re\{[z(1+\epsilon)]^{w/2}\}(1+\hat{\sigma})}{\Re(z^w)(1+\hat{\delta})},$$

where $\hat{\delta}$, $\hat{\sigma}$, and ϵ (because of argument purification) are all real. Again ignoring higher-order terms in the errors, we simplify this to

$$\hat{E} = \hat{\delta} - \hat{\sigma} - \Re(W/2)\epsilon.$$

Similarly,

$$\tilde{E} = \frac{\Im(z^w)(1+\tilde{\delta}) - \Im\{[z^2(1+\epsilon)]^{w/2}\}(1+\tilde{\sigma})}{\Im(z^w)(1+\tilde{\delta})},$$

which simplifies to

$$\tilde{E} = \tilde{\delta} - \tilde{\sigma} - \Im(W/2)\epsilon,$$

is an estimate of the error in the imaginary component.

Table 5 records large errors for routines we suspect use only working-precision arithmetic. The magnitude of these errors is discouraging, but note how often the identity was satisfied exactly. These results demonstrate the difficulty of the computation, and especially how bad things can get for unfortunate argument combinations. Alas, the routines tested here will probably represent the state of the art until an algorithm is found that is as effective and efficient as the algorithm for the real case.

Finally, the test program includes a check with extreme arguments and a check of the special case z^z where $z = 0 + 0i$. All of the programs tested did well with the extreme arguments. In the 0^0 case the Sun programs returned NaNs while the Lahey and VAX programs aborted with invalid argument messages.

9 Test of Complex CSIN/CCOS

Let $z = x + iy$. Then $\sin(z) = \sin(x)\cosh(y) + i\cos(x)\sinh(y)$ and $\cos(z) = \cos(x)\cosh(y) - i\sin(x)\sinh(y)$ are periodic functions of period $2\pi i$ that map infinite vertical strips of width π into the complex plane with slits along the real axis for $|x| > 1$.

Our tests of CSIN and CCOS exploit the identities $\sin(z) = \sin(w)\cos(\Delta) + \cos(w)\sin(\Delta)$ and $\cos(z) = \cos(w)\cos(\Delta) - \sin(w)\sin(\Delta)$, where $w = z - \Delta$ and $\Delta = (1+i)/16$. We minimize subtraction error in these identities by restricting our tests to regions where $y \geq 1/16$ (so the hyperbolic functions are all positive) and where $\sin x$ and $\cos x$ have the same sign and about the same magnitude. Specifically, the test of $\sin z$ is applied over the region with x and y drawn from (1/16,10), and both tests are applied over the region with x and y drawn from (16,17). For these regions, since all arguments in the identities are exact machine numbers, argument purification is not necessary. Finally, numerical stability is enhanced by setting $\tau = \cos(\Delta) - 1$ and computing $\sin(w)\cos(\Delta)$, for example, as $\sin(w) + \tau\sin(w)$. Both $\sin(\Delta)$ and τ are small complex constants that can be precomputed and stored in DATA statements.

The magnitude of the error in CSIN is estimated as the absolute value of

$$E = \frac{\mathrm{CSIN}(Z) - [\mathrm{CSIN}(W)\mathrm{CCOS}(D) + \mathrm{CCOS}(W)\mathrm{CSIN}(D)]}{\mathrm{CSIN}(Z)},$$

where D is Δ. Let δ, σ, and ν denote the relative errors in CSIN(Z), CSIN(W), and CCOS(W), respectively. Then,

$$E = \frac{\sin(z)(1+\delta) - [\sin(w)\cos(\Delta)(1+\sigma) + \cos(w)\sin(\Delta)](1+\nu)}{\sin(z)(1+\delta)}.$$

Retaining only first-order error terms, we reduce this expression to

$$E = \delta - M\sigma - N\nu,$$

where

$$M = \frac{\sin(w)\cos(\Delta)}{\sin(z)}$$

and

$$N = \frac{\cos(w)\sin(\Delta)}{\sin(z)}.$$

For the first test region, x and y each drawn from (1/16,1), $|M| < 0.96$ and $|N| \leq 1$ (see Table 6). Indeed, $|N|$ decreases rapidly as z moves away from $(1+i)/16$, assuming a value of about 0.75 for $z = 1 + i$.

Analysis of the error measurement for the real component is more complicated. Set

$$\hat{E} = \frac{\text{REAL}\{\text{CSIN}(Z)\} - \text{REAL}\{\text{CSIN}(W)\text{CCOS}(D) + \text{CCOS}(W)\text{CSIN}(D)\}}{\text{REAL}\{\text{CSIN}(Z)\}}.$$

Let $\hat{\delta}$ be the relative error in REAL(CSIN(Z)), σ_r and σ_i be the relative errors in the real and imaginary components of CSIN(W), respectively, and ν_r and ν_i be the relative errors in the components of CCOS(W). The usual substitution and simplification yield

$$\hat{E} = \hat{\delta} - \hat{M}_r\sigma_r + \hat{M}_i\sigma_i - \hat{N}_r\nu_r + \hat{N}_i\nu_i,$$

where

$$\hat{M}_r = \frac{\Re\sin(w)\Re\cos(\Delta)}{\Re\sin(z)},$$

$$\hat{M}_i = \frac{\Im\sin(w)\Im\cos(\Delta)}{\Re\sin(z)},$$

$$\hat{N}_r = \frac{\Re\cos(w)\Re\sin(\Delta)}{\Re\sin(z)},$$

and

$$\hat{N}_i = \frac{\Im\cos(w)\Im\sin(\Delta)}{\Re\sin(z)}.$$

The analogous expressions for the relative error in the imaginary component are

$$\tilde{E} = \tilde{\delta} - \tilde{M}_r\sigma_r - \tilde{M}_i\sigma_i - \tilde{N}_r\nu_r - \tilde{N}_i\nu_i,$$

where

$$\tilde{M}_r = \frac{\Re\sin(w)\Im\cos(\Delta)}{\Im\sin(z)},$$

Table 6: **Bounds on Error Magnification for CSIN/CCOS Tests**

		Test	
	SIN	SIN	COS
	$x, y \in (1/16, 1)$	$x, y \in (16, 17)$	$x, y \in (16, 17)$
$\max \lvert M \rvert$	0.96	0.94	0.08
$\max \lvert N \rvert$	1.00	0.08	0.94
$\max \lvert \hat{M}_r \rvert$	0.96	0.92	0.20
$\max \lvert \hat{M}_i \rvert$	0.04	0.01	0.07
$\max \lvert \hat{N}_r \rvert$	1.00	0.20	1.14
$\max \lvert \hat{N}_i \rvert$	0.04	0.06	0.01
$\max \lvert \tilde{M}_r \rvert$	0.09	0.01	0.06
$\max \lvert \tilde{M}_i \rvert$	1.01	1.14	0.20
$\max \lvert \tilde{N}_r \rvert$	1.09	0.07	0.01
$\max \lvert \tilde{N}_i \rvert$	0.09	0.20	0.92

$$\tilde{M}_i = \frac{\Im \sin(w) \Re \cos(\Delta)}{\Im \sin(z)},$$

$$\tilde{N}_r = \frac{\Re \cos(w) \Im \sin(\Delta)}{\Im \sin(z)},$$

and

$$\tilde{N}_i = \frac{\Im \cos(w) \Re \sin(\Delta)}{\Im \sin(z)}.$$

The magnitude of the error in CCOS is estimated as the absolute value of

$$E = \frac{\mathrm{CCOS}(Z) + [\mathrm{CSIN}(W)\mathrm{CSIN}(D) - \mathrm{CCOS}(W)\mathrm{CCOS}(D)]}{\mathrm{CCOS}(Z)}.$$

The expressions for E, $\hat{E}$, and $\tilde{E}$ are the same as for the tests of CSIN, except that now

$$M = \frac{\sin(w)\sin(\Delta)}{\cos(z)},$$

$$N = \frac{\cos(w)\cos(\Delta)}{\cos(z)},$$

$$\hat{M}_r = \frac{\Re \sin(w) \Re \sin(\Delta)}{\Re \cos(z)},$$

$$\hat{M}_i = \frac{\Im \sin(w) \Im \sin(\Delta)}{\Re \cos(z)},$$

Table 7: **Test Results for CSIN/CCOS**

Test/Machine		Vector Error			Error in Real Component			Error in Imag. Component	
	N	MRE	RMS	N	MRE	RMS	N	MRE	RMS
sin(z) **vs** $sin(w+\delta)$ $w = z - \delta, \delta = (1+i)/16$									
$z \in (1/16,1) \times (1/16,1)$									
Sun 3/60 Cal./SP	1104	0.98	0.00	1445	1.25	0.00	1510	1.00	0.00
Sun 3/60 DP	558	1.72	0.12	1016	1.99	0.17	1064	1.99	0.09
IBM XT, Lahey Cal./SP	1135	0.98	0.00	1466	1.40	0.00	1548	0.99	0.00
IBM XT, Lahey DP	1156	0.93	0.00	1526	1.00	0.00	1521	0.99	0.00
VAX, Cal.	1103	0.98	0.00	1498	1.79	0.00	1470	1.00	0.00
VAX, SP	296	3.38	0.70	916	2.04	0.26	625	4.93	1.68
VAX, G-DP	306	3.24	0.68	956	2.22	0.28	613	4.77	1.72
$z \in (16,17) \times (16,17)$									
Sun 3/60 Cal./SP	1088	0.95	0.00	1518	1.00	0.00	1458	1.11	0.00
Sun 3/60 DP	557	1.90	0.12	1062	1.96	0.13	1015	1.92	0.16
IBM XT, Lahey Cal./SP	1101	0.94	0.00	1527	1.00	0.00	1461	1.11	0.00
IBM XT, Lahey DP	1142	0.95	0.00	1556	1.00	0.00	1492	1.00	0.00
VAX, Cal.	1061	0.95	0.00	1513	1.00	0.00	1421	1.11	0.00
VAX, SP	506	1.78	0.17	996	1.89	0.18	983	2.09	0.19
VAX, G-DP	385	1.92	0.33	918	2.17	0.33	849	2.20	0.34
cos(z) **vs** $cos(w+\delta)$ $w = z - \delta, \delta = (1+i)/16$									
$z \in (16,17) \times (16,17)$									
Sun 3/60 Cal./SP	1085	0.96	0.00	1440	1.00	0.00	1520	0.99	0.00
Sun 3/60 DP	525	1.74	0.13	1034	2.09	0.15	994	1.98	0.16
IBM XT, Lahey Cal./SP	1102	0.96	0.00	1437	1.00	0.00	1540	0.99	0.00
IBM XT, Lahey DP	1104	0.95	0.00	1441	1.00	0.00	1539	0.99	0.00
VAX, Cal.	1106	0.96	0.00	1444	1.00	0.00	1535	1.00	0.00
VAX, SP	504	1.79	0.18	954	1.97	0.23	1026	1.87	0.15
VAX, G-DP	394	1.87	0.32	890	2.28	0.35	897	2.09	0.32

$$\hat{N}_r = \frac{\Re\cos(w)\Re\cos(\Delta)}{\Re\cos(z)},$$

$$\hat{N}_i = \frac{\Im\cos(w)\Im\cos(\Delta)}{\Re\cos(z)};$$

$$\tilde{M}_r = \frac{\Re\sin(w)\Im\sin(\Delta)}{\Im\cos(z)},$$

$$\tilde{M}_i = \frac{\Im\sin(w)\Re\sin(\Delta)}{\Im\cos(z)},$$

$$\tilde{N}_r = \frac{\Re\cos(w)\Im\cos(\Delta)}{\Im\cos(z)},$$

and

$$\tilde{N}_i = \frac{\Im\cos(w)\Re\cos(\Delta)}{\Im\cos(z)}.$$

Bounds on the magnitudes of all of these values for the three test domains are given in Table 6. Examination of the table shows that potentially two extraneous rounding errors contaminate test results in the first test region, but that at most one contaminates the other test results. We expect, therefore, that the MRE reported in calibration runs and tests of routines using higher-precision arithmetic internally will be about 1.00, and that other MRE values will be about 2.00. The tabulation of results in Table 7 generally confirms this expectation. Although the errors reported for the first test on the VAX are large, they are not particularly alarming.

The test program concludes with a series of short tests of the periodicity of CSIN and of the response to extreme or near-extreme real and imaginary components of the argument. All routines checked passed these tests; they all accepted large real components of the arguments without complaint, and all gave error returns for large imaginary components (the value $\infty + i\infty$ on the Sun, an underflow message on the PC, and an overflow message on the VAX).

Acknowledgments

This work was motivated by numerous requests for a complex version of the ELEFUNT package and by recent efforts of the Ada Numerics Working Group to draft an Ada standard for complex elementary functions. The author is especially grateful to Peter Tang for being a sympathetic sounding board for ideas and to Gail Pieper for her usual competent editing assistance.

References

[1] *American National Standard Programming Language FORTRAN. ANSI X3.9-1978.* American National Standards Institute, Inc., New York, 1978.

[2] Cody, W. J. MACHAR: A subroutine to dynamically determine machine parameters. *ACM Trans. on Math. Soft. 14* (1988), 303–311.

[3] Cody, W. J., and Waite, W. *Software Manual for the Elementary Functions.* Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[4] Karpinski R. PARANOIA: A floating-point benchmark. *BYTE 10, no. 2* (1985).

[5] Liu, Z. A. Berkeley elementary function test suite, M.S. Thesis, Computer Science Division, Dept. of Electrical Engineering and Computer Science, Univ. of California at Berkeley, Dec. 1987.

[6] Tang, P. T. P. Accurate and efficient testing of the exponential and logarithm functions. *ACM Trans. on Math. Soft. 16* (1990), 185–200.

END

DATE FILMED

02 / 15 / 91